AD-A251 644

(2)

A Scheduling Approach to Verifying Safety in Real-Time Systems*

Armen Gabrielian
UniView Systems
1192 Elena Privada, Mountain View, CA 94040
armen@well.sf.ca.us

1992

Abstract

A general method for reasoning about the behavior of real-time systems is presented using "hierarchical multi-state (HMS) machines." The method is based on a parametric scheduling approach that reduces the problem of verification of safety properties of concurrent systems to the solution of a set of inequalities. The method is applied to obtain a very simple solution to the problem of safety of a real-time concurrent mutual exclusion protocol that has been studied by a number of authors recently.

Keywords — Real-time systems, safety property, specification, verification, scheduling, HMS machines, temporal logic, automata.

1 Introduction

Numerous methods for the specification and verification of real-time systems have been proposed in the recent past. The choice of method is often based on esthetic grounds and there are no generally agreed-upon criteria for judging specification and verification methods. In addition, it is difficult to establish links between various techniques because of differences in how system behavior is represented, what form the constraints or requirements take, how concurrency is defined, what model of time is employed, what syntactic notation is used, and what approaches to verification are employed.

A useful tool for establishing a common frame of reference among various specification and verification methods is the use of common examples. However, it should be pointed out that relative judgments about different methods cannot be made from a few examples. The

*This work was supported in part by the Office of Naval Research under contract N00014-92-C-0047.

main advantage is to improve understanding of various approaches and to allow consideration of problems from different points of view.

Recently, a mutual exclusion protocol attributed to Michael Fischer has been proposed as a paradigmatic simple example of concurrent real-time behavior [ALP91]. The safety property of this protocol has been studied in [AL91] using "temporal logic of actions" [AL91] and in [SBM91] using the notion of "proof outline." In the former the proof is omitted due to its length (four pages [ALP91]) and in the latter only part of the proof of a slightly more general version is presented.

The purpose of this paper is to introduce a general method of verifying safety properties of a class of real-time systems and to demonstrate how the method can be applied to obtain a very simple proof of safety of the mutual exclusion protocol discussed above. An important benefit of our method is that the solution of the problem is reduced to the satisfaction of a set of inequalities which are relatively straightforward to evaluate. It is hoped that our work will foster further dialogue among proponents of various verification methods for real-time systems and will make the field of formal verification accessible to a larger audience.

The specification method used in this paper is based on an integration of parallel and hierarchical automata and a temporal interval logic, called "hierarchical multi-state (HMS) machines," that has been studied extensively during the last few years [GF88, FG89, GF91, Ga91a, GI91, GI92, Ga91b]. Fundamental ideas of HMS machines are as follows: (1) state behavior is represented in terms of automata in which multiple states can be true at a moment of time, (2) multiple transitions can fire in the automata simultaneously, (3) enablement conditions of transitions are defined in terms of predicates on histories of states and transitions, expressed in a temporal interval logic called TIL or a variation thereof, and (4) a state may consist of an HMS machine, including a copy of itself (giving rise to recursive hierarchies). While most of the research on HMS has been for discrete time, recent work has extended the basic concepts to the continuous-time domain [Ga91b]. Also, a general algebraic framework for embedding "objects" in states was introduced in [GI90] that allows very compact modeling of systems containing many similar entities such tracks, plots or elements of a real-time database. In this paper, we present a very simple version of this concept to illustrate the reduction in complexity that is achieved by the use of objects in states. As far as the verification of properties of real-time systems specified by HMS machines is concerned, five methods have been developed so far, some of which are interrelated.

Two of the HMS machine verification methods address the verification of a safety property by extending an HMS machine representing a real-time system with a new "system failure (SF)" state. The state SF has the property that it becomes true if and only if the safety property is violated, and once it does becomes true, it can never become false again. Thus, to prove safety, it is sufficient to demonstrate that SF is unreachable, since SF is false originally. To this end, in [FG89] a correctness-preserving transformation method was presented which permits the repeated modification of the structure of an HMS machine representing a real-time system in such a way that (1) the behavior of the specification remains unchanged, and (2) if the safety property is satisfied in the specification, the corresponding SF state becomes isolated, thereby demonstrating its unreachability. The second verification method

in this class, presented in [GI91, Ga91b], employs a model-based theorem proving approach using an extension of the semantic tableau method [Fi90] to demonstrate a safety property through a refutation-based argument.

In [GI92] preliminary results on model checking for HMS machines were presented. The key difference with the traditional model checking approach [CES86] is in the nature of the computation graphs generated. In particular, in the computation graph of HMS machines, (1) a node can contain not just the label for a single state that may be true, but *histories* of arbitrary finite numbers of states expressed in a subset of TIL, (2) edges may denote the firing of multiple transitions,(3) finite delays when no transitions fire are represented conveniently in a parametric form, and (4) nondeterminism has a somewhat more liberal interpretation. In [GI91] a method of decomposing a computation graph into a number of smaller interactive parametric computation graphs for a special class of commonly-occurring HMS machines was introduced. In general, this reduces the size of the computation graph enormously and permits a convenient analysis of many properties of interest using simple algebraic arguments.

The fifth and final previous verification method for HMS machines to be considered here was introduced in [GF91] and is inherently tied to nondeterminism which is defined somewhat differently in HMS machines from the standard notion in automata theory and Petri nets. Standard nondeterminism arises out of structural considerations when more than one transition is enabled from a state (or place in the case of Petri nets). For HMS machines, each transition is labeled as deterministic or nondeterministic, with the interpretation that an enabled deterministic transition will fire and a nondeterministic transition *may* fire. Thus, multiple transitions from a single transition may fire simultaneously. Also, nondeterminism is a convenient mechanism for defining arbitrary delays on transitions. The scheduling-based verification method of [GF91] assumes that a "plan" or sequence of transitions with arbitrary delays for each transition is given. By a symbolic execution of such a plan, one can create a set of integer inequalities that will determine if a "schedule" for the plan can be found to satisfy all the temporal and logical constraints in the specification.

The verification approach to be presented here combines a number of the previous techniques presented above in a more general context. In particular, it employs continuous-time HMS machines and integrates ideas from the interactive parametric computation graphs of [GI91] with a nonsequential version of the scheduling method of [GF91].

In Section 2 of this paper we present an overview the basic concepts of HMS machines and in Section 3 we give an outline of our verification method and illustrate it with a simple example. In Section 4 we apply our verification method to the mutual exclusion algorithm discussed above. In Section 5 we present the conclusions.

2 Background and Definitions

We begin by introducing an informal definition of HMS machines. We note that the main difference between our definition here with most of previous work on HMS machines (as exemplified, e.g., by [Ga91a]) is in the use of continuous time instead of discrete time. All

other concepts are essentially identical in the two types of time domains.

Definition 1 *A continuous-time "hierarchical multi-state (HMS) machine" is a triple* $H = (S, \Gamma_D, \Gamma_N)$*, where*

1. *S is a set of "states," any number of which may be true or "marked" at a moment of time.*

2. *Γ_D and Γ_N are, respectively, the sets of "deterministic" and "nondeterministic" transition of H. Each deterministic or nondeterministic transition is of the form*

$$(PRIMARIES)\ (CONTROL) \rightarrow (CONSEQUENTS),$$

where $PRIMARIES \subseteq S$, $CONSEQUENTS \subseteq S$, and CONTROL is a predicate on the history of the states and the transitions expressed in a temporal interval logic called C-TIL. For a transition u, each state in the associated PRIMARIES (CONSEQUENTS) set of state will be called a "primary" ("consequent") state of u. Also, the predicate CONTROL for u will be called the "control" or "control predicate" of u.

3. *A transition is "enabled" at time t if its primary states are all true at the "premoment" of t (see Definition 2) and its associated control predicate is true at t.*

4. *At each (continuous) moment of time, all the enabled deterministic transitions and a subset of the nondeterministic transitions "fire."*

The hierarchical version of HMS machines arises when a state is another HMS machine itself. An HMS machine is "recursive" if it contains a copy of itself as a state. For the purpose of this paper, a hierarchy can be considered as a grouping of states. Further details about hierarchies, including discussion about the use of different clocks at different levels, can be found in [Ga91a, GI92].

The dynamic behavior of a critical real-time system can be specified in terms of an HMS machine by representing its attributes as states, with the control predicates defining the logical and temporal constraints under which changes in the system must or can occur. Extensive arguments on the advantages of HMS machines over traditional finite-state machine or Petri net representation schemes can be found in [Ga91a]. Principal advantages are in reduced number of states, hierarchical decomposition, accommodation of different granularities of time, representation of systems at multiple levels of abstraction, and a richer language for defining temporal constraints. In addition, there exist a variety of methods for verifying a very general form of (non-probabilistic) "safety" that, at least in the hard real-time case, subsumes various concepts of dependability such as guaranteed service, fault-tolerance and even security.

We now present a notation and a formal definition for the temporal interval logic C-TIL, the continuous version of the discrete-time interval-based temporal logic TIL which has been studied extensively in previous work on HMS machine theory. A slight variation of C-TIL

was introduced in [Ga91b]. As evidenced by [Ko90, NS91], interest in the use of continuous time for modeling real-time systems has grown considerably in recent years. One additional change from most of the previous work on HMS machines is that, following the approach of [FG89], we include in our definition "transition-based" control predicates in C-TIL that simplify the definition of synchronization constraints on the firings of transitions.

Notation *Given an HMS machine $H = (S, \Gamma_D, \Gamma_N)$, the "marking" of H at time t is a mapping $M_t : S \cup \Gamma_D \cup \Gamma_N \rightarrow \{F, T\}$ that defines the set of marked or true states of H and the transitions that fire at time t.*

Definition 2 *Given a marking M_t of an HMS machine at time t and a formula ψ, we denote the satisfiability of ψ in M_t by $M_t \models \psi$. C-TIL then in obtained by extending propositional logic with the following five operators:*

$\mathbf{O}(t')$	At *relative time* t'
	$M_t \models \mathbf{O}(t')\psi \Leftrightarrow M_{t+t'} \models \psi$
$[t_1, t_2]$	Always *from* t_1 *until* t_2
	$M_t \models [t_1, t_2]\psi \Leftrightarrow \forall t'\ t_1 \leq t' < t_2$ *implies* $M_t \models \mathbf{O}(t')\psi$
$\mathbf{O}(t'^-)$	At premoment *of relative time* t'
	$M_t \models \mathbf{O}(t'^-)\psi \Leftrightarrow \exists \varepsilon > 0$ *such that* $[t - \varepsilon, t]\psi$
$<t_1, t_2>$	Sometime *from* t_1 *until* t_2
	$M_t \models <t_1, t_2> \psi \Leftrightarrow \exists t'$ *such that* $t_1 \leq t' < t_2 \wedge M_t \models \mathbf{O}(t')\psi$
$<t_1, t_2>!$	Sometime-change *from* t_1 *until* t_2
	$M_t \models <t_1, t_2>! \psi \Leftrightarrow \exists t'$ *such that*
	$((t_1) \leq t' < t_2) \wedge (M_t \models \mathbf{O}(t'^-)\neg\psi) \wedge (M_t \models <t', t_2> \psi)$.

Definition 3 *For each state s in an HMS machine, let $\Gamma_{in}(s)(\Gamma_{out}(s))$ be the set of transitions into (out of) s. Then, the marking of s at time t is defined as follows:*

$$\mathbf{O}(t)s \Leftrightarrow (\mathbf{O}(t^-)s \wedge (\textstyle\bigwedge_{u \in \Gamma_{out}(s)} \mathbf{O}(t)\neg u)) \vee (\textstyle\bigvee_{v \in \Gamma_{in}(s)} \mathbf{O}(t)v),$$

where for a function ψ, $\bigwedge_{x \in X} \psi(x) = T$ if $X = \{\}$ and $\bigvee_{x \in X} \psi(x) = F$ if $X = \{\}$.

Intuitively, a state s is true at time t if and only if (1) s is true for a non-zero interval ending at t, where the interval is closed on the left and open on the right, and no transitions fire out of it at t, and/or (2) some transition fires into s at time t. A transition is true at time t if and only if it fires at t.

Figure 1 presents our graphic notation for a a simple HMS machine that will be used in the next section to introduce our scheduling-based verification method. In our notation, boxes represent states, dark arrows denote transitions with an asterisk indicating that the transition is nondeterministic, thin arrows represent controls, and temporal operators appear next to the symbol Ⓣ. VLSI notation is in general used to form logical combination of control predicates and multiple controls on a single transition represent the conjunction of the individual predicates. Thus, the nondeterministic transition x in Figure 1 may fire at time t if $\mathbf{O}(t^-)A$ and $[-2, 0]A \wedge [-3, 2]C \wedge [-1, 0]D$.

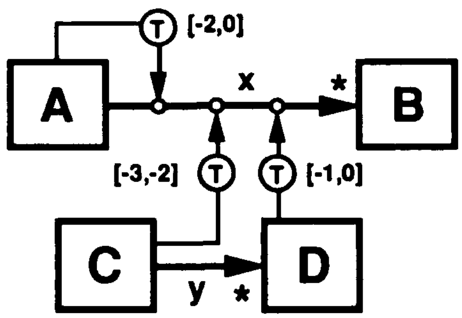

Figure 1: A Simple HMS Machine Denoting a Parallel Scheduling Problem

3 A Scheduling-Based Verification Method

Given an HMS machine with nondeterministic transitions such that of Figure 1, one can define a "plan" as a sequence of sets of transitions that can fire one after the other, with delays between successive firings denoted by terms of the form ϕ^i, where ϕ represents *no action* and i is the length of the delay. For example, for Figure 1, a possible plan is $p = \phi^i y \phi^j x$, which consists of a wait of i time units, followed by the firing of the transition y, followed by a wait of j time units, followed by the firing of the transition x. By finding a solution for i and j that satisfies all the control predicates for y and x, one obtains a "schedule" for p. In [GF91], a general method was presented in the discrete time domain for determining choices for the parameters i and j that would results in feasible solutions for p. The demonstration of the infeasibility of the plan p proves safety if p is the only path to an unsafe state.

The scheduling method of [GF91] is based on a parametric execution of plans that compares the constraints defined by control predicates to parametric facts that can be deduced during the execution of the plan. This method was used there to derive schedules for the operation of a steam generator monitoring system. The details of the method will not be repeated here.

In this paper, we consider three modifications of the scheduling method of [GF91]: (1) continuous time is used instead of discrete time, (2) multiple concurrent plans are employed, resulting in significant reduction in complexity for many applications, and (3) the evaluation of facts during the execution of plans are simplified by adopting a variation of lazy evaluation. The basic ideas will now be illustrated for the example of Figure 1.

We assume that initially the states A and C are marked in the HMS machine of Figure 1 and we wish to find a schedule for the firing of the transitions x and y that would satisfy the controls on x. We consider the pair of parametric plans $\phi^\alpha x$ and $\phi^\beta y$. The term ϕ^α (ϕ^β) in the first (second) plan defines a delay of α (β) before the firing of x (y).

According to the scheduling method of [GF91], which can be generalized to the continuous case for multiple plans, at each step of the execution of a parametric plan, we compare known facts with the required facts, the former derived by the execution of the plan and the latter from the control predicates of the transitions in the plan. The basic idea is that if for a

state s we know that $[\alpha_1, \beta_1]s$ is true and we also have a requirement that $[\alpha_2, \beta_2]s$ *must* be true, then we can conclude that $[\alpha_2, \beta_2] \subseteq [\alpha_1, \beta_1]$, where brackets are used simply to represent intervals. From this we can derive the two inequalities $\alpha_1 \leq \alpha_2$ and $\beta_1 \geq \beta_2$. One additional improvement on the application of this concept to be considered here is that instead of considering *all* the known facts as in [GF91], here we limit our analysis to only *relevant* facts in the spirit of lazy evaluation.

The transition y in Figure 1 has the null control predicate and does not impose any constraints on the parameters α and β. When the transition x fires at relative time zero in the plan $\phi^{\alpha}x$, we have the following facts using the rules of [GF91]:

$$[-\alpha, 0]A \wedge [-\alpha, -\alpha + \beta]C \wedge [-\alpha + \beta, 0]D$$

We now consider the control predicate $[-2, 0]A \wedge [-3, -2]C \wedge [-1, 0]D$ of the transition x. By comparing the known facts with the three terms in this predicate, we can derive a set of inequality constraints on the delays α and β. The the relevant inequalities derived for each term of this control predicate are underlined below:

$$\begin{array}{ll}
[-2,0]A\text{:} & [-2,0] \subseteq [-\alpha,0] \Rightarrow \\
 & \underline{\alpha \geq 2} \\
[-3,-2]C\text{:} & [-3,-2] \subseteq [-\alpha,-\alpha+\beta] \Rightarrow -\alpha \leq -3,\ -\alpha+\beta \geq -2 \Rightarrow \\
 & \underline{\alpha \geq 3},\ \underline{\alpha \leq \beta+2} \\
[-1,0]D\text{:} & [-1,0] \subseteq [-\alpha+\beta,0] \Rightarrow -\alpha+\beta \leq -1 \Rightarrow \\
 & \underline{\alpha \geq \beta+1}
\end{array}$$

From underlined inequalities we immediately conclude that $\beta \geq 1$, $\alpha \geq 3$, and $\beta + 1 \leq \alpha \leq \beta + 2$. Thus, we obtain a complete set of permissible schedules for the execution of the transitions y and x. The same principle will be used in the next section to determine the conditions under which one can construct a feasible schedule for a set of plans that leads to the violation of a safety property.

4 Verification of a Mutual Exclusion Protocol

We now consider the application of the scheduling-based verification method of the last section to the problem of verifying a mutual exclusion protocol attributed to Michael Fischer. As mentioned in the Introduction, this protocol has been studied in [AL91], while a slightly more general version has been considered in [SBM91]. This protocol consists of a number of processes such that each process i executes the following code:

$$\begin{array}{rl}
a\text{:} & \textbf{await } \langle x = 0 \rangle; \\
b\text{:} & \langle x := i \rangle; \\
c\text{:} & \textbf{await } \langle x = i \rangle; \\
\text{cs:} & \text{critical section}
\end{array}$$

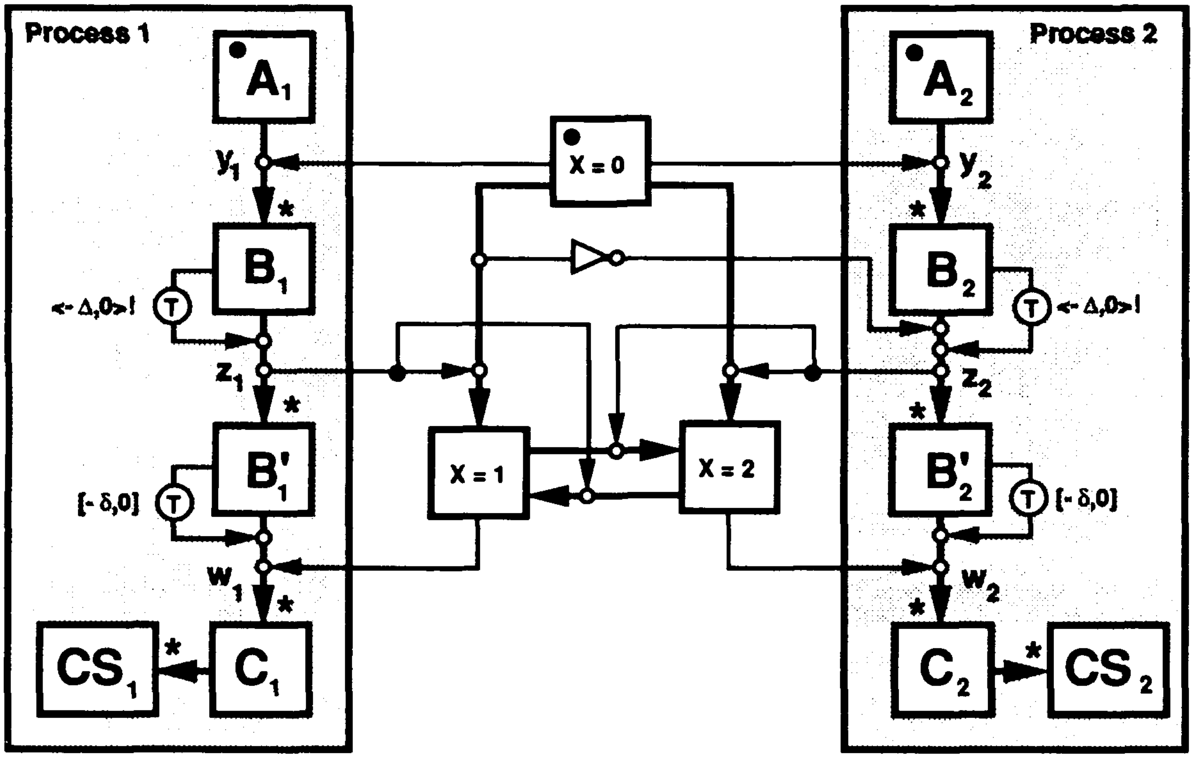

Figure 2: Two Interacting Processes in Fischer's Mutual Exclusion Protocol

In each process i there is a maximum delay Δ between the execution of the test in statement a and the assignment in statement b, and a minimum delay δ between the assignment in statement b and the test in statement c. The safety property we want to verify is the following: what conditions on the parameters Δ and δ will guarantee that only one process can be in its critical section *cs* at one moment of time. As noted in [ALP91], the protocol is incomplete since it allows lockout of processes. This can easily be remedied, however, and it does not affect the proof of safety.

We can formalize the protocol for two processes, which we call Process 1 and Process 2, in terms of the HMS machine of Figure 2. It is easy to see that it is sufficient to prove safety for just two processes. The state A_i corresponds to statement a in the list of statements for process i above, states B_i and B_i' correspond to statements b, and so on. All the transitions in the two processes are nondeterministic (indicated by the asterisks next to the arrowheads), permitting arbitrary delays between successive step. On the other hand, the transitions among the global state $x = 0$, $x = 1$, and $x = 2$ are all deterministic. Initially, states A_1, A_2, and $x = 0$ are marked as indicated with the solid small circles. Considering Process 1, y_1 can fire as long as $x = 0$ is true. The transition z_1 can fire if B_1 has *become true* during the last Δ time units (indicated by the *sometime-change* operator), and if $x = 0$ is true. Similarly, w_1 can fire if B_1' has been true for at least δ time units and $x = 1$ is true. The negative control on transition z_2 is merely a device to prevent the simultaneous firing of z_1

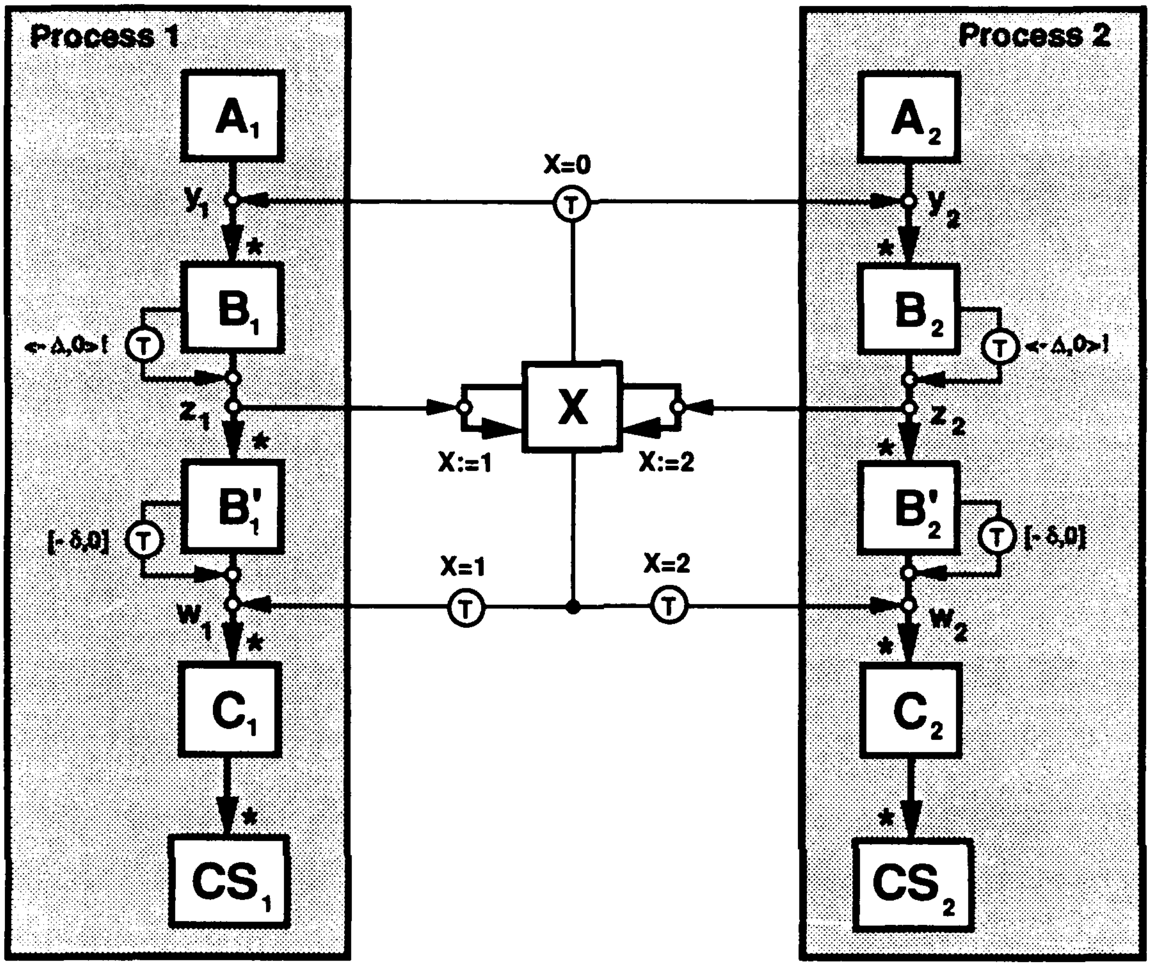

Figure 3: Specification of Fischer's Mutual Exclusion Protocol with Objects in States

and z_2.

Figure 3 presents an alternate and simpler HMS machine specification of the Fischer's protocol using "objects" in states following [GI90]. In this case, the state X actually contains the variable X that can be modified as transitions fire from X. We also assume an interleaving operation of this specification in order to avoid the need for an additional control as in Figure 2. The following proof can be applied to both Figures 2 and 3.

We consider the possibility that, starting from the initial condition, both critical sections become true - actually, we only need to assume that C_1 and C_2 become true. This implies that the following two plans must both be feasible:

$$p_1 = \phi^{\gamma_1} y_1 \phi^{\alpha_1} z_1 \phi^{\beta_1} w_1$$
$$p_2 = \phi^{\gamma_2} y_2 \phi^{\alpha_2} z_2 \phi^{\beta_2} w_2$$

Since w_1 and w_2 cannot fire simultaneously, without any loss of generality, we assume that p_1 completes before p_2. We can then derive the following inequalities from the constraints

on the transitions:

(1)	$\gamma_1 + \alpha_1 + \beta_1 \leq \gamma_2 + \alpha_2$	Since when w_1 fires z_2 has not fired yet.
(2)	$\gamma_2 \leq \gamma_1 + \alpha_1$	Because, otherwise, y_2 cannot fire.
(3)	$\alpha_1 \leq \Delta$	By definition.
(4)	$\beta_1 \geq \delta$	By definition.
(5)	$\beta_2 \geq \delta$	By definition.
(6)	$\alpha_2 \leq \Delta$	By definition.

Adding the inequalities (1) and (6), we get $\gamma_1 + \alpha_1 + \beta_1 \leq \Delta + \gamma_1$. Rewriting (4) as $\delta \leq \beta_1$ and adding it to the sum of (1) and (6), we get $\gamma_1 + \alpha_1 + \delta \leq \Delta + \gamma_2$. Adding this to (2), gives us $\delta \leq \Delta$, which is a necessary condition for the violation of the desired safety property. Therefore, to guarantee that the safety property will not be violated, it is sufficient to have $\Delta < \delta$. This is identical to the safety condition stated in [AL91].

5 Conclusions

In this paper, a scheduling-based method for verifying safety properties of real-time systems specified in terms of "hierarchical multi-state (HMS) machines" was presented. The method was applied to obtain a very simple proof of safety of a mutual exclusion protocol that has been studied by several authors using other techniques. An important advantage of our method is that proof of safety is reduced to the solution of a set of inequalities which are, in general, relatively easy to evaluate. Since the notion of safety is rather general, the method can be applied to verify the trustworthiness and dependability of real-time systems from a wide set of perspectives.

References

[AL91] Abadi, M., and L. Lamport, "An old-fashioned recipe for real time," *Proc. 1991 REX Real-Time Workshop*, Springer-Verlag, 1992 (to appear).

[ALP91] Abadi, M., L. Lamport, and V. Pratt, Several informal comments on Concurrency Net, November 1991.

[CES86] Clarke, E.M., E.A. Emerson, and A.P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic," *ACM Transactions on Programming Languages and Systems,* Vol. 8, No. 2, 1986, pp. 244-246.

[Fi90] Fitting, M., *First-Order Logic and Automated Theorem Proving*, Springer-Verlag, 1990.

[FG89] Franklin, M.K., and A. Gabrielian, "A transformational method for verifying safety properties in real-time systems," *Proc. 10th Real-Time Systems Symposium,* Santa Monica, CA, December 5-7, 1989, pp. 112-123.

[Ga91a] Gabrielian, A., "HMS machines: a unified framework for specification, verification and reasoning for real-time systems," *Foundations of Real-Time Computing: Formal Specifications and Methods*, A. van Tilborg and G. Koob (Eds.), Kluwer Academic Publishers, Norwell, Mass., 1991, pp. 139-166.

[Ga91b] Gabrielian, A., "Verifying safety properties of HMS machine specifications by (inductive) theorem proving (extended abstract)," *Proc. Fourth Annual Review and Workshop, Foundations of Real-Time Computing Research Initiative*, Washington, DC, Oct. 31 - Nov. 1, 1991, pp. 111-116.

[GF88] Gabrielian, A., and M. K. Franklin, "State-based specification of complex real-time systems," *Proc. IEEE Real-Time Systems Symposium*, Huntsville, AL, 1988, pp. 2-11.

[GF91] Gabrielian, A., and M.K. Franklin, "Multi-level specification real-time systems," *Communications of the ACM*, Vol. 34, No. 5, May 1991, pp. 50-60. Earlier version in *Proc. 12th International Conference on Software Engineering*, March 26-30, 1990, Nice, France, pp. 52-62.

[GI90] Gabrielian, A., and R. Iyer, "Specifying real-time systems with *extended* hierarchical multi-state (HMS) machines," Technical Report No. 90-21, Thomson-CSF, Inc., Palo Alto Reserch Operations, January 1990.

[GI91] Gabrielian, A., and R. Iyer, "Verifying properties of HMS machine specification of real-time sytems," *Proc. Third Workshop on Computer-Aided Verification,* Aalborg, Denmark, July 1-4, 1991, pp. 489-500. Revised version to appear in Springer-Verlag LNCS series.

[GI92] Gabrielian, A., and R. Iyer, "Integrating automata and temporal logic: a framework for specification of real-time systems and software," in *The Unified Computation Laboratory*, C.M.I. Rattray and R.G. Clark (eds), Institute of Mathematics and Its Applications, Oxford University Press, March/April 1992, pp. 261-273 (to appear).

[Ko90] Koymans, R., "Specifying real-time properties with metric temporal logic," *Real-Time Systems,* Vol. 2, No. 4, November 1990, pp. 255-299.

[NS91] Nicollin, X. and J. Sifakis, "An overview and synthesis on timed process algebras," *Proc. Third Workshop on Computer-Aided Verification,* Aalborg, Denmark, July 1-4, 1991.

[SBM91] Schneider, F.B., B. Bloom, and K. Marzullo, "Putting time into proof outlines," Technical Report No. TR91-1238, Dept. of Computer Science, Cornell University, Ithaca, NY, September 1991.